NASA

Ames Research Center

PROPOSAL FOR CONSTRUCTING AN ADVANCED SOFTWARE TOOL FOR PLANETARY ATMOSPHERIC MODELING

Richard M. Keller, Michael H. Sims, Esther Podolak
Christopher P. McKay and David E. Thompson

Report RIA-90-03-20-1
March, 1990

RIA

Artificial Intelligence
Research Branch

Proposal for Constructing an Advanced Software Tool for Planetary Atmospheric Modeling

March 1990

Richard M. Keller Michael H. Sims Esther Podolak

Christopher P. McKay David E.Thompson

NASA Ames Research Center
Mail Stop 244-17
Moffett Field, CA 94035

(415) 604-3388 (Keller) (415) 604-4757 (Sims) (415) 604-6997 (FAX)
Keller/Sims@pluto.arc.nasa.gov or MHSims (NASAMail)

Abstract

Scientific model-building can be a time-intensive and painstaking process, often involving the development of large and complex computer programs. Despite the effort involved, scientific models cannot easily be distributed and shared with other scientists. In general, implemented scientific models are complex, idiosyncratic, and difficult for anyone but the original scientist/programmer to understand. We believe that advanced software techniques can facilitate both the model-building and model-sharing process. We propose to construct a *scientific modeling software tool* that serves as an aid to the scientist in developing and using models. The proposed tool will include an interactive intelligent graphical interface and a high-level domain-specific modeling language. As a testbed for this research, we propose development of a software prototype in the domain of planetary atmospheric modeling.

Table of Contents

1. Introduction

1.1. Challenges in Scientific Model-building

Model-building is an integral part of all scientific enterprise. Scientists studying a particular phenomenon develop theories in order to account for novel observations and to make predictions about expected behavior. To validate their theories, scientists conduct *in vivo* experiments whenever possible. Often, however, it is impracticable to carry out such direct experiments due to cost or other limiting factors. For example, often psychologists and biologists are unable to conduct human experiments due to ethical concerns; space scientists are unable to conduct planetary experiments due to logistical factors; theoretical physicists are unable to conduct experiments due to hardware technology limitations. In these cases, scientists build models of the system under study (e.g., a human brain, a planetary atmosphere, an atom) and then test their theories against those models. Sometimes these models take the form of hardware (i.e., some sort of physical analog to the actual system), but often models are expressed in software. We focus on software models in this proposal.

Scientific model-building can be a time-intensive and painstaking process. Many scientific models are written in terms of numeric programming languages, such as FORTRAN. Implementing a model can involve writing large and complex programs that access multiple datasets and utilize numerous different statistical and numeric processing packages. Development time for large scientific models can take on the order of many months to years of effort.

For all the time and effort it takes to develop a model, the user community for most scientific models is limited to one -- the scientist who initially designed it. This is not to say that one scientist has no use for another's models. On the contrary, model-sharing is highly desirable because it gives scientists the ability to "run experiments" and test their theories using different models without additional development overhead. The ability to easily inspect, use, and modify another scientist's models would be an extremely useful adjunct to the model-building process, and an effective medium for communicating scientific ideas, as well. Just as scientists read technical papers describing theories, they should be able to inspect and exercise the software models that were used in validating those theories. Given the benefits to model sharing, why is it practiced so infrequently?

There are numerous technological barriers to scientific model sharing.[1] Some of these barriers include:

- *Lack of comprehensibility*: As with most research software, scientific models are often sparsely commented and difficult to understand. Even a program that is initially well-written and commented becomes increasingly fragmented and cryptic as inevitable changes to the model are made over time.
- *Wrong level of abstraction:* The structure of the scientific model is not obvious from the low-level implementation that confronts the would-be user of the model. FORTRAN or other general programming languages do not include the scientific terms and concepts that are natural to the scientist. As a result, the model-builder is forced to translate a model originally expressed on paper in terms of natural scientific concepts into a "foreign language" of 'constants', 'variables', 'subroutines', and 'looping constructs'.
- *Unmodifiability:* Due to problems with comprehensibility and level of abstraction, models are often difficult to change to suit the needs of a different scientist.

[1] There are also sociological barriers to model sharing that evolve from the way in which institutions reward scientists for results. Scientists are not typically rewarded for building a piece of software that helps other scientists achieve results. In fact, they are discouraged from sharing resources that might enable others to make a discovery before themselves.

- *Implicit assumptions:* Often important modeling assumptions and data assumptions are left implicit in the low-level code that implements the model. These implicit assumptions cannot be easily inspected or modified by a potential new user. This is a significant deterrent to using another scientist's model because the appropriateness of assumptions is frequently the source of scientific debate.

Not surprisingly, some of these barriers are quite similar to those cited as discouraging traditional software sharing [Cheatham 84, Green *et al.* 83].

1.2. Objectives and Approach

Our primary research objective is to facilitate scientific model-construction and model-sharing by addressing the technological barriers described above. Despite the fundamental nature of scientific modeling, there is little software support available for this important activity. We propose to investigate the development of specialized software tools to ease the modeling process.

In particular, we believe that the following collection of advanced software techniques can substantially enhance the modeling process. We propose to integrate these techniques in a *scientific modeling software tool* that serves as an aid to the scientist in developing and using models. The techniques include:

- *Interactive graphical interface*: To enhance comprehensibility and modifiability of models. Visual and iconic representations help the user to rapidly grasp the content of a model.
- *High-level modeling language*: To provide an appropriate level of abstraction for modeling and introduce natural domain concepts that are familiar to the scientist-user.
- *Analysis facilities*: To facilitate interpretation of experimental results through use of graphical plotting and statistical techniques.
- *Intelligent assistance*: To provide guidance and automate simple modeling steps. Artificial Intelligence-based techniques, such as constraint satisfaction, typed inheritance hierarchies, and backward-chaining control can reduce the amount of detail that the scientist-user needs to track.
- *Assumption maintenance facility*: To maintain explicit descriptions of modeling and data assumptions underlying a model and interdependencies among these assumptions.

Rather than attempting to construct a general-purpose modeling tool, our approach is to first focus narrowly on a particular scientific domain and a specific class of models within that domain. Because the nature of scientific models differs widely across different scientific disciplines, we believe that software support can have the greatest impact by maintaining a narrow focus. Once we have some depth of experience in one scientific domain, we can widen and generalize our software techniques to accommodate other modeling problems within that domain and other similar domains.

We have developed a software prototype of a scientific modeling tool in the domain of planetary atmospheric modeling to test the feasibility of this idea. The next section of the proposal describes this domain and our initial prototype. We consider the results of this unfunded feasibility study to be promising, and the intent of this proposal is to obtain funding to continue this research and to enhance the prototype currently developed.

2. Application to Planetary Atmospheric Modeling

2.1. Atmospheric Modeling

The focal point for our preliminary research has been a model of the thermal and radiative structure of Titan's[2] atmosphere. This model is implemented in a FORTRAN program (consisting of over 5000 lines of code) which was developed by co-investigator McKay [McKay *et al.* 89]. The model was designed with two principal goals in mind: first, to be a general tool for the investigation of scientific issues associated with Titan, and second, to provide a mechanism for the generation of synthetic spacecraft data, either for use in the design of future instruments or for the analysis of data from previous instruments. Since the model is meant to be a general tool for the study of Titan's atmosphere, it has submodels that represent all the basic atmospheric physics. It includes solar radiation, thermal infrared radiation, methane and ethane clouds, organic haze microphysics, real gas thermodynamics, and comparisons with Earth-based and Voyager datasets.

The McKay model is in great demand by planetary scientists -- both as a resource in support of ongoing Titan research and as an aid to study and planning activities associated with NASA's upcoming Cassini/Huygens Probe to Titan. In addition to simply running the Titan model "as is", planetary scientists frequently have the need to create variations of this basic model. These variants incorporate different underlying assumptions about Titan's atmosphere, leading to different calculations in certain portions the model. For example, the current model incorporates a relatively simple 2-stream model of radiation transfer within Titan's atmosphere. However, to study more subtle radiation transfer problems, scientists need to incorporate a more sophisticated n-stream model. Currently, the original programmer (McKay) is the only one who can make this type of modification with any degree of facility.

Our intention is to make the Titan model available for use by a larger group of planetary scientists. Specifically, the objective of our prototype is to permit scientists to inspect an atmospheric model, modify parts of the model, execute the model, and perform analyses on the results. Rather than augment the existing FORTRAN Titan model, our approach has been to build a interactive program that enables a user to reimplement McKay's original program using a high-level atmospheric modeling language that sits at a level of abstraction far above the FORTRAN code level. The model is automatically compiled into lower-level code by the system before it is executed. The terms in the atmospheric modeling language involve domain concepts that are familiar to the scientist, including physical variables ("saturation point"), physics equations ("ideal gas law"), and experimental datasets ("Voyager refractivity data"). By interacting with a graphical interface, users essentially "program" using this more natural high-level modeling language. Available FORTRAN subroutines may be utilized as 'black boxes' when appropriate.

Our initial prototype focuses on the gaseous composition portion of the Titan model, which is described in the next section.

2.2. Modeling Titan's Gaseous Composition

The purpose of the gaseous composition portion of the overall model is to develop a profile of Titan's atmosphere that describes the pressure, temperature, and density of gases at various altitudes above its surface. This problem is underconstrained due to the shortage of empirical data on Titan. The major source of relevant experimental data is the Voyager I flyby of Titan back in November 1980. As Voyager I passed by the far side of Titan, it sent back radio waves that passed through Titan's atmosphere and then on to receiving stations on Earth. Due to the density of gases in the atmosphere,

[2]Titan is one of Saturn's moons.

the radio waves were refracted slightly as they passed through the atmosphere. The amount of refraction was measured at different altitudes above the surface. This refractivity data serves as a starting point for inducing the desired atmospheric profile in the Titan gas composition model [Lindal *et al.* 83].

To summarize, the atmospheric profile is determined as follows (see Figure 1). First, for each atmospheric point profiled, the Voyager I refractivity data is used to compute the number-density (ND) of the gases at that altitude. The number-density of a mixture of gases is defined as the number of molecules per volume of the mixture. Assuming the identity and relative percentages of gases in a mixture is known[3], the number-density can be computed as a function of refractivity. Next, using the molecular weight of the various gases in that mixture, the mass-density (RHO: mass per volume of mixture) can be computed from the number-density. The hydrostatic law can then be used to determine the pressure (P) from the mass-density by essentially summing the weight of the atmosphere above each profile point. Finally, the temperature (T) can be determined from the mass-density and the pressure by applying an equation of state, such as the ideal gas equation.

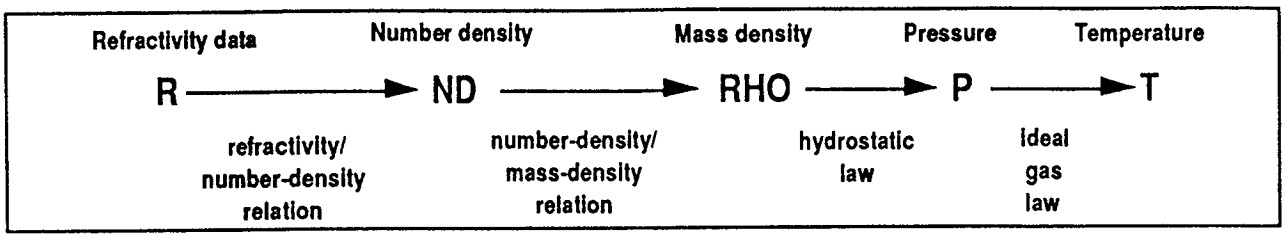


Figure 1: Determining the atmospheric profile

Figure 1 illustrates a level of abstraction at which a physicist might describe the problem, and is far more comprehensible than the corresponding FORTRAN code. Our goal is to construct a graphical interface and an associated modeling tool with which a physicist can construct a model with this level of abstraction.

By retrospectively reconstructing the development of the Titan model, we have determined that the basic schema for computing an atmospheric profile was refined several times during the course of model construction. The different refinements stem from different modeling assumptions about the identity and percentage of gases in the atmosphere. Initially, a single gas (N_2) was assumed. Then the model was altered to accommodate multiple gases (N_2, Ar, CH_4, H_2) at fixed percentages. Finally, the percentage of each gas was made variable as a function of altitude. This actual sequence of model modifications serves an important role as a driver for our research. As a proof of concept, we are attempting to build into our modeling tool the functionality that would enable a user to easily replicate this sequence of model construction and model modification steps.

2.3. Prototype

In this section, we describe our current prototype for the atmospheric modeling tool. It is this prototype which we intend to extend under the proposed funding. For the purposes of our initial prototype, we have conceptualized model-building as a process of linking uncomputed physical variables (or data structures such as vectors) to computed variables using *computational transformations*. For example, the process of linking the input Voyager I refractivity data to the output ideal gas temperature is accomplished by the sequence of transformations illustrated in Figure 1. Conceptually, each transformation takes as input a set of variables and produces a single variable as output. Physics

[3]The actual gas composition of Titan's atmosphere is uncertain, but scientists can make some reasonable assumptions based on empirical data obtained from Voyager I instruments and earth-based telescopes.

equations and subroutines are two kinds of computational transformations supported in the current system. Physics forms the basis for atmospheric modeling, so the use of physics equations is natural in this context. The introduction of subroutines is motivated by the fact that in building models, scientists often makes use of program code that has been developed elsewhere. Examples range from standard numeric integration and data interpolation packages to complex scientific models developed by other scientists. The details of this code are assumed to be outside the scope and interest of the scientist-user. Subroutines are an effective method of incorporating these 'externally-supplied' transformations, yet shielding the user from necessary but uninteresting details in the modeling process.

In our prototype, the process of linking of variables using transformations is accomplished via a simple backchaining procedure. In this backchaining process, the user first selects a target physical variable they wish to calculate. Then the system presents the user with a set of transformations that can be used to compute this variable. The user selects one of these transformations, and the system checks to see whether all the input variables used by this transformation have already been determined. If so, the target variable is computed using the transformation; if not, the backchaining process recurses for each of the undetermined variables. This process will be illustrated further below.

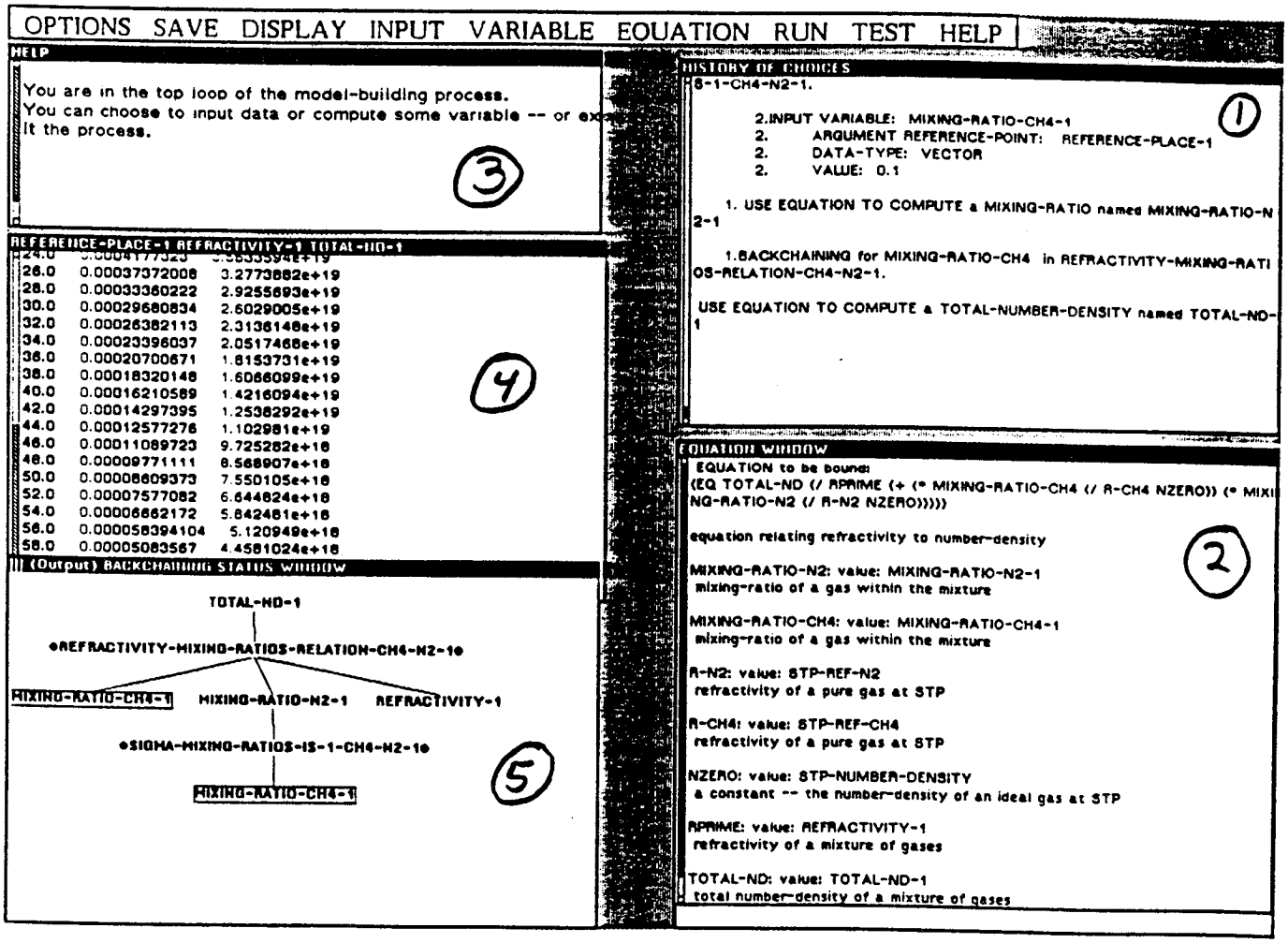


Figure 2: Standard modeling tool display screen

2.3.1. Tool Functionality

The standard display screen of the current interactive modeling tool is illustrated in Figure 2. The screen features a menu bar of pull-down menus (as used in a Macintosh interface) across the top and a variety of windows for displaying relevant information during the model-building process.

The functions of the various pull-down menus are as follows:

- **Variable menu**: Enables the user to select the next physical variable (e.g., temperature, pressure) to be computed in the model under construction.
- **Equation menu**: Enables the user to select a transformation (i.e., equation or subroutine) to apply to the last computed variable.
- **Display menu**: Enables the user to visualize the model and its associated variables in a variety of formats, including plotted graphs and data tables for displaying computed variables, and a global tree-like structure for displaying dependencies among the variables in the current model.
- **Input menu**: Enables the user to select an experimental dataset for inclusion in the current model, as well as to define and modify user input variables for the model.
- **Run menu**: Enables the user to execute the current model after modifications have been made.
- **Test menu**: Enables the user to apply one or more user-defined tests of model viability to the results produced by the current model. These tests provide feedback to the user on the credibility of the current model. For example, a one of the tests applied to the Titan models is a test for atmospheric stability. If the temperature gradient predicted by the model is too steep, the atmosphere is inherently unstable, which goes against normal expectations. If a model fails a test of viability, the scientist may choose to modify the current model until it passes the test.
- **Help menu**: Provides informative system-level user assistance
- **Options menu**: Provides various model-management functions. Each scientist using the modeling tool has a separate workspace in which to store models. Within a single workspace, the scientist can store several related models. The options menu allows the user to switch the current focus of the model-building activity to a different model and/or workspace, and to initiate new models or delete existing models.
- **Save menu**: Enables user to save the current model in the current workspace.

The windows shown in Figure 2 are used in the following manner:

- **History window** (#1 in Figure 2): A scrolling window that sequentially displays each modeling step executed during the current model-building session.
- **Equation window** (#2): When a transformation (i.e., equation or subroutine) is selected for application by the user, it is displayed in the equation window for purposes of verification and for reference during the subsequent equation instantiation process.
- **Help window** (#3): Provides informational help to the user during the model-building process.
- **Display window** (#4): Displays tables, graphs, or trees when requested by the user via the display menu.
- **Backchaining status window** (#5): Displays the status of any currently-active backchaining process. This window is updated dynamically after each step of the backchaining.

2.3.2. Model Construction Scenario

To give the flavor of how models are built and extended using the tool, this section describes a portion of a model construction scenario in some detail. For the purposes of this discussion, we will assume that the user has already built an atmospheric model, and wishes to extend the model. In particular, we will assume the scientist has already constructed a model that determines pressure from refractivity data (see Figure 1's first 3 steps). Now the scientist wishes to extend the model to compute temperature, as well. This addition is accomplished with the following steps:

- Step 1: The user recalls the incomplete model stored in his or her workspace using the **options menu**. A visualization of the current model is then displayed to the user (see Figure 3). The tree-like structure shows the transformations applied thus far in the model, as well as the associated input and output variables, and their interdependencies. For example, reading down from the root of the tree, we see that the last variable computed was *P-1*, a pressure variable. *P-1* was computed using an instance of the hydrostatic law (*hydrostatic-law-1*) based on the input values *G-1* — the gravitational force at a particular point in the atmosphere — and *RHO-1* — the mass-density at that point. *RHO-1* was in turn computed from an equation relating number-density and mass-density (*density-from-nd-1*). The input number-density (*total-nd-1*) was computed using the relationship between number-density, refractivity, and the relative percentages of each of the gases in the atmosphere (e.g., *mixing-ratio-ch4-1*). The variables at the leaves of the tree are primitive inputs to the model, and are assumed to be given. (Note that the boxed variables in the tree appear more than once in different places.)
- Step 2: The user selects *temperature* from the **variable menu** (see Figure 4) indicating she or he next wishes to compute temperature within the model.
- Step 3: In response to the user's selection of a physical variable, the system displays a menu of all transformations that can be applied to compute that variable (see Figure 5). This is the beginning of a backchaining process initiated to compute *temperature*.

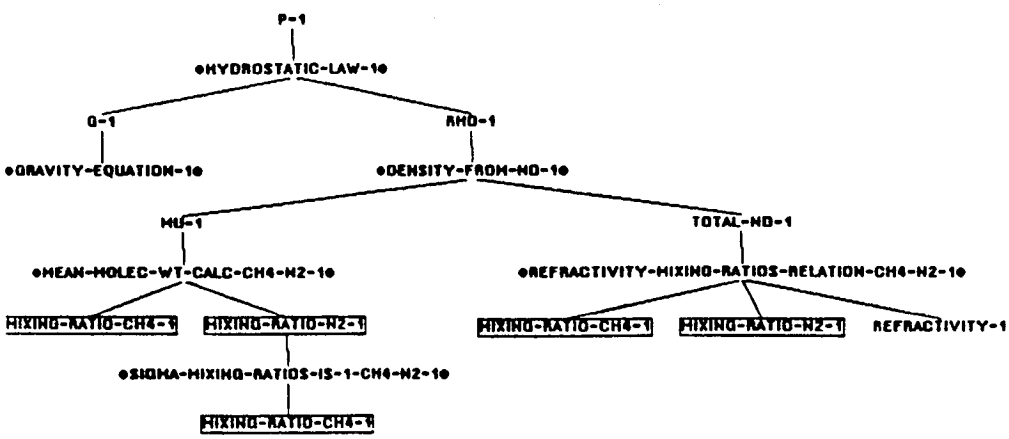
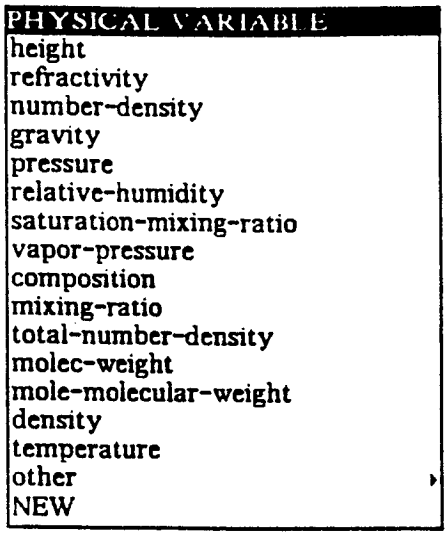


Figure 3: Dependency-tree visualization of incomplete atmospheric model



Figures 4: Variable menu

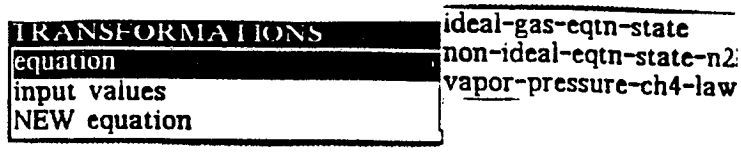


Figure 5: Transformation menu

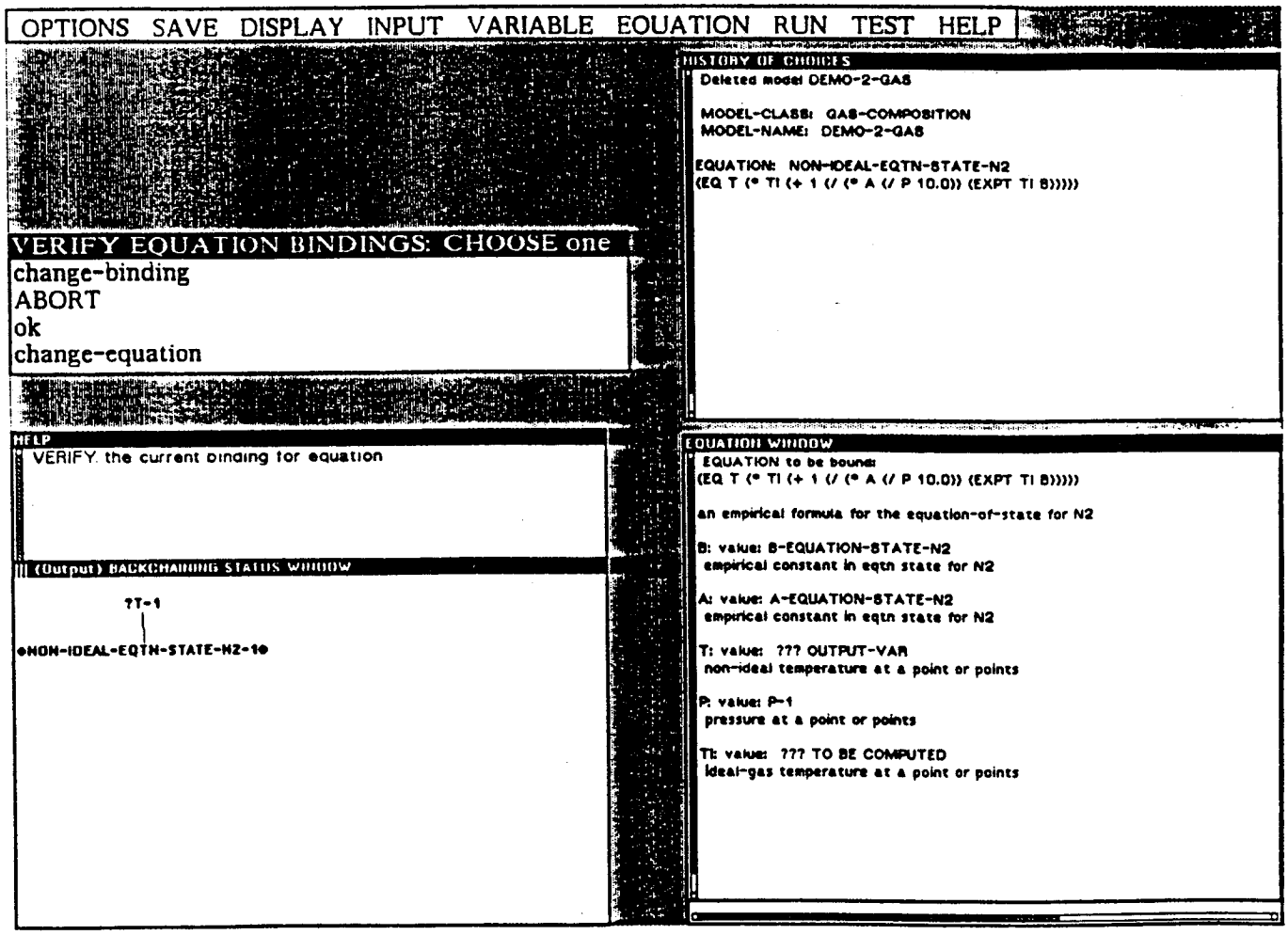


Figure 6: Tool display screen at beginning of interaction

- Step 4: The user chooses one of the transformations -- in this case, a non-ideal gas equation of state. (This non-ideal gas equation corrects the ideal gas temperature using an empirically-derived correction factor.) The user's choice is echoed in the **history window**, and the system displays the transformation in the **equation window** for verification (see Figure 6). In the **equation window**, the complete equation is displayed along with the symbols used and their meanings. At this point, if the user has made an incorrect choice, the interaction can be aborted and different choice made. The system makes a preliminary attempt to bind the symbols in the equation to known variables in the current model, and these bindings are displayed in the **equation window** subject to user approval.

  Simultaneously, the system displays the status of the backchaining process in the **backchaining status window** (Figure 6). Here the temperature variable to be computed (*T-1*) is shown as an output of an instance of the non-ideal gas equation of state (*non-ideal-eqtn-state-n2-1*). The "?" indicates that the value of the *T-1* variable has yet to be computed.

- Step 5: Because there are two temperature symbols in the non-ideal gas equation (*TI* , an ideal gas temperature, and *T*, the corrected non-ideal gas temperature), the user is asked to select one of these symbols as the output to the equation. The system will manipulate the equation so that the symbol to be computed is on the left hand side of the equal sign in the equation.

- Step 6: After the bindings have been verified and the output variable has been selected, the system updates the **backchaining status window** to display the selected bindings (Figure 7). Note that at this point, the new *temperature* variable (*T-1*) has been connected to the original incomplete model (shown in Figure 3) by means of the instantiated pressure variable *P-1*, which was introduced by the non-ideal gas equation of state. Because the ideal gas temperature (*TI-1*) has yet to be computed, the system now begins recursively backchaining on this variable.

- Step 7: As in Step 3, the system prompts the user to select the transformation to be used in computing the ideal gas temperature. This time the user picks the ideal gas equation, rather than the non-ideal gas equation. The inputs to this equation are number-density and pressure, which have already been computed as part of the original incomplete model. The system binds these variables and the backchaining process terminates. The system now percolates the data values from the leaves of the tree up to the root to compute first the ideal gas temperature (*TI-1*), and finally the non-ideal gas temperature (*T-1*).

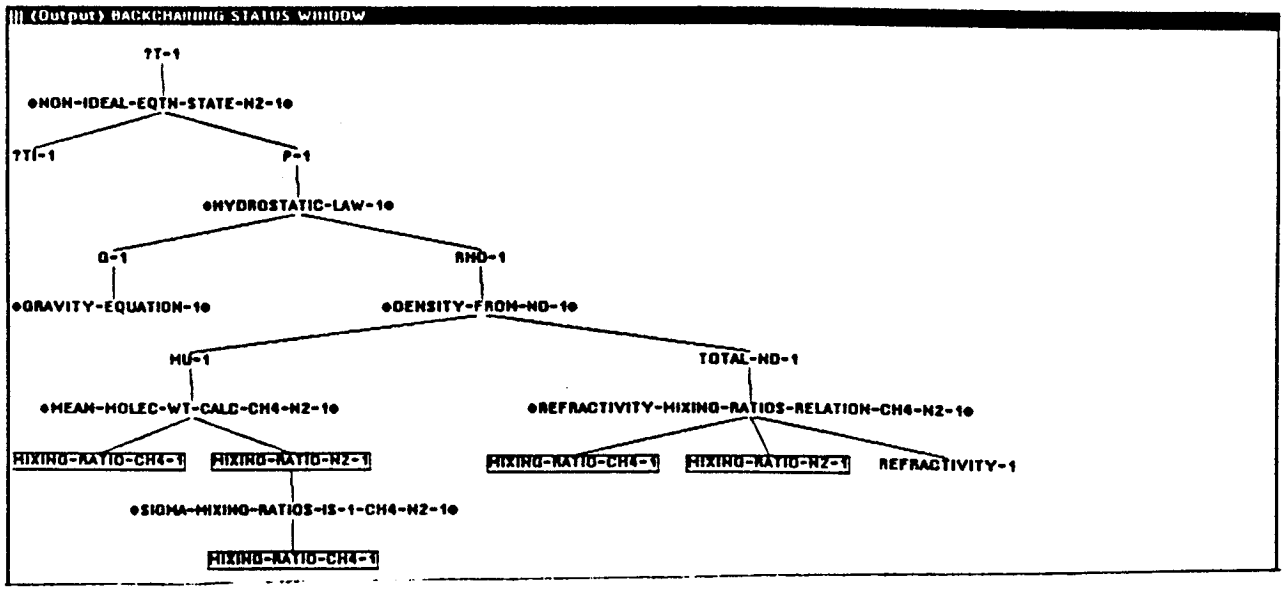


Figure 7: Updated backchaining status window

•Step 8: At this point, the user may choose to plot the value of non-ideal temperature as a function of altitude (Figure 8) by making an appropriate selection in the **display menu.** Or the user may decide to apply a test of model viability from the **test menu.** For example, the *saturation law* specifies that the mixing ratio of a given gas in the atmosphere must be less than the saturation mixing ratio for that gas. In other words, gases are not expected to supersaturate under normal circumstances. The test selected by the user appears in the equation window (Figure 9a). The system applies the test and reports whether it is satisfied by the results produced by the current model. In this case, the test fails, as is illustrated by the plot of mixing ratio and saturation mixing ratio versus altitude (Figure 9b), which the user has requested. Note in the region between 5 and 50 kilometers above the surface, the saturation mixing ratio (dark line) is less than the 5% fixed mixing ratio (light vertical line) that the user has defined in the model.

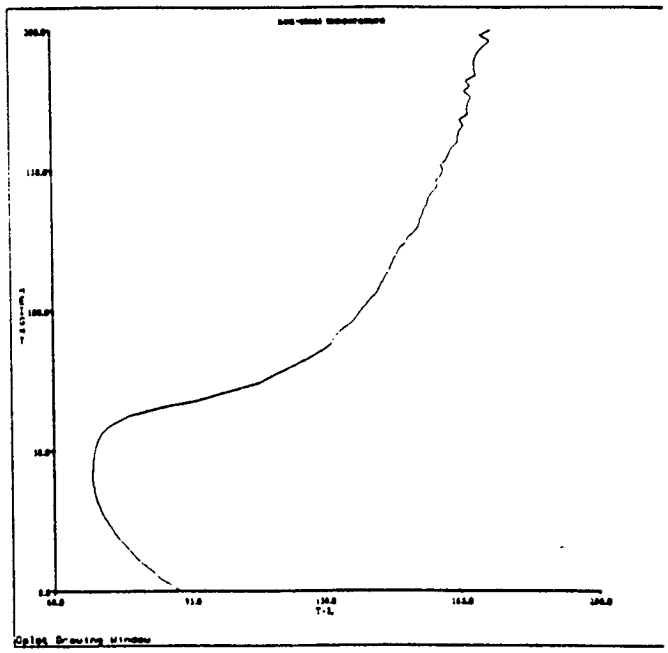


Figure 8: Plot of non-ideal temperature versus altitude

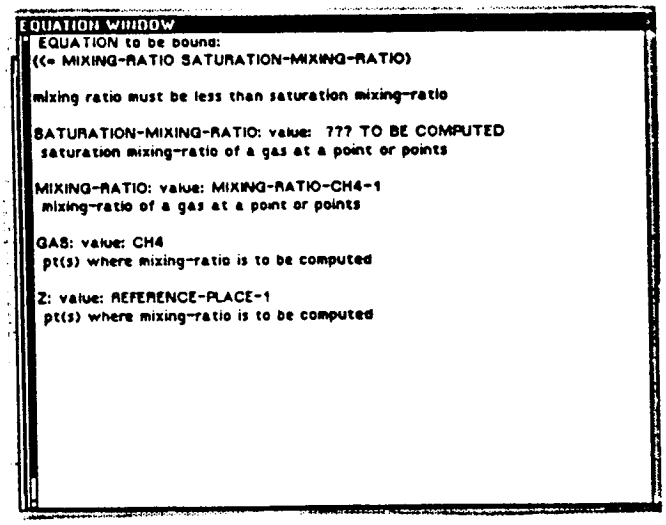


Figure 9a: Saturation law

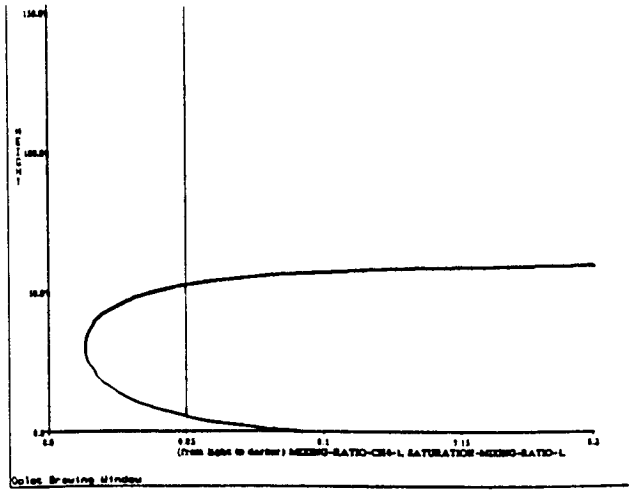


Figure 9b: Plot of mixing ratio (light line) and saturation mixing ratio (dark line) versus altitude

2.3.3. Status

To date, we have focused on model construction rather than model modification, although the tool supports simple types of changes to existing models. The modeling tool can be used to construct two of the three variants of the basic gas composition model described at the end of Section 2.2 above (the 1-gas and the Fixed-percentage/multi-gas variants). This partial prototype has been implemented in LISP using Intellicorp's object-oriented knowledge representation tool KEE running on a Symbolics workstation. This development environment currently meets our needs for rapid prototyping. We plan on moving to Sun Sparcstations because the Symbolics and KEE are on temporary loan, and Sparcstations offer a more economical option. We envision eventual movement to a better delivery platform, such as a Macintosh.

3. Research Issues: Lessons from the initial prototype

The construction of our partial prototype has given concrete shape to the abstract concept of an atmospheric modeling tool. Our experience thus far in building and testing the prototype has pointed out a number of issues that must be addressed to facilitate the use of this tool by atmospheric scientists. In this section, we discuss some of these research issues.

3.1. Knowledge Representation

The quality of the tool's interaction with a planetary scientist depends critically on the system's knowledge about the modeling task and the related atmospheric physics. If the system had a greater understanding of atmospheric modeling, it would be capable of providing an additional level of support for the modeling task. Specifically, additional knowledge would enable the system to present the user with more intelligent choices at decision points. By making "intelligent choices", we mean the system should provide better system defaults, better ordering of user options, and a reduced (i.e., filtered) set of options.

There are several places where more intelligent choices are desirable in the current system. For example, the system currently displays a nearly exhaustive list of physical variables and transformations when the user clicks on the variable or equation menus, respectively. As the number of transformations and variables increases, this exhaustive display will cease to be a viable option. The system must intelligently filter the set of transformations or variables that are relevant in the current modeling context.[4]

Another place where the system must make intelligent choices is in binding equation symbols to physical variables. For example, when the user chooses to apply the ideal gas equation to compute temperature, the system must bind the symbols for pressure and number-density to previously-computed physical variables. If a number of different pressure and number-density variables have been computed in the current model, the system must determine which of these variables the user intends to bind to the equation symbols. If the system "understands" the ideal gas equation, it can apply certain constraints to ensure necessary consistency among its choices. For instance, to apply the ideal gas equation properly, the selected pressure and number-density variables must pertain to the same gas mixture at the same location. Any attempt to apply the equation to a pressure at the Titan's surface and a number-density at 200 meters above the surface should be recognized and rejected as nonsensical. This type of constraint is normally implicit in the computer encoding of the model. It is precisely this implicit encoding that

[4]Actually, the system currently performs some intelligent transformation and variable filtering. For example, during backchaining, the system filters the set of transformations displayed so that only those involving the backchained variable are shown.

makes the creation, modification, and sharing of these programs with colleagues so difficult. Explicit representation of domain knowledge and reasoning processes is an approach to solving these problems.

An atmospheric scientist brings a variety of different kinds of knowledge to bear in approaching the modeling task. This includes both domain knowledge, such as knowledge about physical variables, physics equations, and the structure of atmospheres, and modeling knowledge, such as various strategies and heuristics for atmospheric modeling, and various constraints and assumptions pertaining to the modeling process. We believe that representing knowledge of this type is the key to developing a useful modeling assistant. To represent these different types of knowledge, we propose using a combination of various knowledge representation techniques from Artificial Intelligence (AI), including object-oriented, frame-based, logic-based, and teleologic representations, in addition to inheritance hierarchies and constraints [Brachman & Levesque 85].

3.2. Interface

The interface is a critical part of the modeling tool because it serves as an intermediary between the scientist and the system. No matter how capable the tool is, it will not be useful unless the scientist can easily and naturally interact with the system -- both in communicating information to the system and in obtaining information from the system. The modeling task presents some special problems in both regards.

When scientists interact with each other, they share a common body of implicit knowledge and assumptions which facilitates the communication of ideas. In contrast, currently when a scientist communicates with the computer, implicit knowledge must be explicitly communicated, and this can make the interaction cumbersome and tedious. For example, when a scientist refers to the refractivity of a pure gas, it is assumed (unless otherwise stated) that the gas is under STP (standard temperature and pressure) conditions. We would like the system to share this implicit assumption and relieve the user of the burden of stating it explicitly. Similarly, when a scientist builds an atmospheric model, he or she may fail to specify certain low-level data transformations that are -- strictly speaking -- required. For instance, input data may require interpolation or its units may require conversion. In these cases, we would like the system to perform the necessary low-level transformations without insisting that the user fully specify these actions. Of course these transformations should be available for subsequent inspection by the user.

To address the problem of communicating implicit information, we envision the need for a translation process that augments the user's incomplete modeling directives into unambiguous, full-specified directives by adding necessary implicit knowledge. This implicit knowledge would be derived from appropriate defaults and contextual information, and would be fully accessible to the user if needed. Research on natural language referent resolution is relevant to this problem [Allen 88].

Another important interface issue involves the design of model presentation and visualization aids. The model-building process is complex and the resulting models are complicated artifacts. We have not yet developed wholly adequate methods for displaying to the user the current model and the associated model-building steps. At present, we make use of a scrolling display in the history window to document the model-building steps, and a mouse-sensitive, tree-like structure to visualize the model itself. This structure displays the interrelationships among variables and the transformations that were used in computing those variables. We will be analyzing the efficacy of these display methods, and will attempt to redesign our interface in accordance with recommendations stemming from research in human-computer interaction [Card *et al.* 83, Schneiderman 87].

3.3. Modeling Assumptions

One of the reasons it is so difficult for scientists to share models is that their modeling assumptions are often implicit in the code that implements the model. For example, here are some of the modeling assumptions buried in the FORTRAN code that implements the Titan model:

- *"CH_4 will condense in Titan's atmosphere; N_2 will not."*
- *"The relative humidity of CH_4 at Titan's surface is 60%."*
- *"The source of CH_4 is at Titan's surface."*
- *"Atmospheric conditions are (or are not) consistent with ideal gas assumptions."*
- *"Titan's temperature profile is no steeper than the adiabatic temperature profile."*
- *"Gases in Titan's atmosphere are well-mixed so that refractivities of individual gases in the atmosphere are additive."*
- *"The wavelength emitted during the Voyager I refractivity experiments is close enough to the wavelength used in developing STP-refractivity indices so that the difference can be ignored.*

One of our goals is to make these types of modeling assumptions explicit and inspectable by other scientific users. A shortcoming of our current prototype is that it has no special facilities for representing or maintaining modeling assumptions. When a scientist makes a modeling change that violates an existing modeling assumption, we want the system to automatically inform the user of the inconsistency. We look toward AI techniques as a basis for handling these problems of assumption maintenance [Doyle 79] and explicit representation of modeling assumptions [Smith *et al.* 85].

3.4. Related Research on Scientific Modeling Tools

There are several pieces of relevant research on the construction of scientific models and modeling tools in various domains. [Abelson *et al.* 89] describes a suite of intelligent programs developed at M.I.T. for automatic preparation, execution, and control of numerical experiments. These programs have been combined in a package called the "Kineticist's Workbench" to produce a sophisticated tool for modeling and analysis of dynamical systems. [Robertson *et al.* 89] and [Bundy *et al.* 90] report on the ECO system developed at the University of Edinburgh to facilitate the construction of ecological simulation models. [Kant 88] describes a system under development at Schlumberger to help scientist perform mathematical model-building for data interpretation tasks such as oil well log interpretation. Finally, [Nielsen 88] presents an analysis of the tool functionality required to make specific classes of modifications to a simulation model of a communications network.

4. Project Plan

Thus far, we have focused our prototyping activity on the gas composition portion of the Titan model. Our project plan for Year 1 (FY91) is geared toward completing our initial prototype and extending its coverage of McKay's model. Year 2 (FY92) continues development of the system in response to scientists' feedback on our initial prototype, and results in a system capable of being used for active experimentation by atmospheric scientists. In particular, researchers (including investigator McKay) are interested in performing certain experiments on gas composition and atmospheric stability using the modeling tool we develop. Although these experiments could be conducted by modifying the existing FORTRAN model, this would be tedious and difficult for scientists including McKay. One of our main short-term goals is to facilitate the conduct of these modeling experiments. In Year 3 (FY93) we would extend the tool and deliver a prototype tool to the atmospheric sciences community at NASA and ESA

for use in conjunction with their instrument-design activities related to the upcoming Cassini mission, which includes the deployment of a Titan probe. During the three-year effort, we intend to fully support the tool's use by the interested scientific community and be responsive to the needs of those users. After gaining experience with the tool, we will investigate the application of these techniques to other planetary domains, as well as to other scientific modeling domains.

4.1. Year 1 (FY91): Completion of initial prototype

- Model analysis: Analyze McKay *et al.*'s Titan model and identify what additional portions of the model should be supported by the initial prototype; determine what types of model changes should be supported;
- Knowledge representation: Continue further development of appropriate representations for domain concepts, physical quantities, physics equations, model constraints, and modeling assumptions;
- Interface: Design graphical interface to modeling tool that facilitates construction and modification of Titan gas composition model;
- Modeling Assumptions: Develop facility for explicitly representing and maintaining modeling assumptions;
- Model modification: Design and implement enhanced support for user modification of existing models;
- FORTRAN interface: Develop the capability to incorporate existing atmospheric models and subroutines written in FORTRAN.

Year 1 deliverable: Initial prototype to be tested and evaluated by atmospheric scientists at Ames

4.2. Year 2 (FY92): Extension/revision of prototype into working assistant

- Enlarge scope of modeling effort: attempt to extend the core modeling system to cover a larger portion of the original FORTRAN model
- Revise graphical interface in response to feedback from initial users;
- Incorporate symbolic equation manipulation/simplification package, such as MACSYMA;
- Develop facility to maintain multiple, competing atmospheric models simultaneously;
- Port first version of system from AI workstation to a more suitable delivery platform (e.g., Mac II);
- Validation: Planetary scientists use model construction tool to conduct numeric experiments on atmospheric composition and stability.

Year 2 deliverable: Refined prototype to be used in conducting atmospheric science experiments

4.3. Year 3 (FY93): Enhancement of tool using advanced software techniques

- Incorporate additional degrees of autonomy into the system, thereby lessening the burden on the human model builders;
- Generalize tool to different modeling domains (e.g., different planetary atmospheres, physical systems other than atmospheres).

Year 3 deliverable: Enhanced system for use in the Cassini instrument design phase

5. Significance for Planetary Atmospheric Research

One of our primary goals is to extend McKay's computer model of Titan [McKay *et al.* 89] using the proposed modeling tool so it better meets the needs of the planetary atmospheres user community. We plan to augment the model as follows. In the initial stages of model extension, there will be an emphasis on the coupling between the atmospheric structure, the haze particles, and the radiatively active molecules in the atmosphere. Radiative transfer in the visible and thermal infrared provides the primary inputs that govern the atmosphere. Photochemical reactions in the atmosphere result in the conversion of CH_4 and N_2 to heavier compounds and haze material. This haze will be carefully modeled since it plays a crucial role in the thermal structure due to the absorption of sunlight. Condensation of organic molecules and the formation of clouds will be included. Coupling of the atmosphere to a possible surface ocean will also be included. The models will consist of layered 1-D models within 2-D models that allow for the computation of zonal wind fields. It is probable that winds play an important role in determining the altitude and latitude distribution of the high altitude haze. In addition to these planned extensions, we intend that microphysics submodels currently under development --- such as the poly-disperse haze model being developed by O. Toon *et al.* (in preparation) --- will be incorporated into the current model by means of the modeling tool.

We have indicated that an important aspect of the proposed investigation is to make the current Titan model more easily accessible to the scientific user community. Hence, a significant effort will be devoted to providing a user interface that will allow the user to choose the combinations of models and output modes that best suit the problem. For example, a user who is primarily interested in collecting haze particles for organic analysis might wish to substitute a complex poly-disperse 2-D aerosol model for the current simple microphysics model. In addition, the user must be able to interrogate, define and vary submodel parameters, such as the total haze production rate. As the Titan model is improved and refined, new versions of the software will be released and distributed to the user community. The modeling tool will be made portable, and individual PI's would be able to generate any desired outputs using the tool at their home institutions.

An important testing ground for the proposed modeling tool is the upcoming Cassini mission -- a joint mission between NASA and the European Space Agency (ESA), whose objective is to explore the Saturn system. This mission will include a Titan probe. The software tool we propose to develop will be of considerable use to researchers responding to Cassini mission requirements. There are several distinct phases in the Cassini mission and we anticipate a corresponding set of uses for the Titan modeling tool. We intend to pace the development of the modeling tool based upon the Titan probe schedule, and we envision that this system will evolve into a flexible tool that will have many uses during all phases of the Cassini mission. During the latter part of the instrument design phase (Oct. 1990 - May 1994) we anticipate that the tool will be available to support a variety of design activities, including generating design requirements for the collecting instruments, selecting filters for optical instruments, determining probable ranges of results for atmospheric measurement instruments, performing sensitivity checks, etc. After the instrument designs have been finalized and before the return of any real data (1994-2001), we expect that the Titan model and the tool will have at least the following two uses. First, we anticipate the tool will aid in simulating Probe descent under varying conditions and assumptions in order to optimize the operations of the instruments during the actual descent within the restrictions of the hardware. Second, we envision that the tool will be used to develop specific modeling procedures and data sets for use in analysis of the anticipated data returned by the Probe. After final acquisition of data during Titan encounter in 2001, the modeling tool will find its ultimate use in the analysis of the actual data.

6. Conclusion

The initial beneficiaries of the software tool we have proposed will be those scientists involved in studies of Titan. However, we believe the software development program we have proposed will provide a good test of the entire concept of intelligent graphics interfaces as an aid in constructing and modifying complex scientific models. If this proves to be feasible, the potential impact is substantial, as there are literally hundreds of fundamentally similar specialized scientific models in a variety of disciplines ranging from atmospheric sciences to biochemistry. Based on our experience in building an atmospheric modeling tool, we hope to generalize our techniques and apply them to other domains of NASA interest. As we stated in the introduction to this proposal, we hope our efforts will go some distance toward breaking down the formidable barriers to scientific model construction and model sharing that face our scientific community today.

7. References

[Abelson *et al.* 89] H. Abelson, M. Eisenberg, M. Halfant, J. Katzenelson, E. Sacks, G.J. Sussman, J. Wisdom, and K. Yip, "Intelligence in Scientific Computing", *Communications of the ACM*, 32, 546-562, May 1989.

[Allen 88] J. Allen, *Natural Language Understanding*, Benjamin/Cummings, Menlo Park, CA, 1988.

[Bundy *et al.* 90] A. Bundy, R. Muetzelfeldt, D. Robertson, and M. Uschold, "Computer-Aided Design of Program Specifications in the Domain of Ecological Modelling", University of Edinburgh Department of AI technical report, 1990.

[Card *et al.* 83] S.K. Card, T.P. Moran, and A. Newell, *The Psychology of Human Computer Interaction*, Lawrence Erlbaum, Hillsdale, NJ, 1983.

[Cheatham 84] T.E. Cheatham, Jr., "Reusability Through Program Transformations", *IEEE Transactions on Software Engineering*, 10:5, pp. 589-594, September 1984.

[Doyle 79] J. Doyle, "A Truth Maintenance System". *Artificial Intelligence*, vol. 12, no. 3, pp. 231-272, 1979.

[Green *et al.* 83] C. Green, D. Luckham, R. Balzer, T. Cheatham, and C. Rich, "Report on a Knowledge-Based Software Assistant", Kestrel Institute technical report, June 1983.

[Kant 88] E. Kant, "Approximation and Learning in Mathematical Modeling", *Proceedings of the AAAI Spring Symposium on Explanation-Based Learning*, Palo Alto, CA, 1988.

[Lindal *et al.* 83] G.F. Lindal, G.E. Wood, H.B. Hotz, D.N. Sweetnam, V.R. Eshleman, and G.L. Tyler, "The Atmosphere of Titan: An analysis of the Voyager I radio occultation measurements", *Icarus*, **53**, 348-363, 1983.

[McKay *et al.* 89] C.P. McKay, J.B. Pollack, and R. Courtin, "The Thermal Structure of Titan's Atmosphere", *Icarus*, **80**, 23-53, 1989.

[Nielsen 88] N.R. Nielsen, "The Application of AI Techniques to Model Maintenance", SRI International technical report, 1988.

[Robertson *et al.* 89] D. Robertson, A. Bundy, M. Uschold, and R. Muetzelfeldt, "The ECO Program Construction System: Ways of increasing its representational power and their effects on the user interface", *Int. J. of Man-Machine Studies*, 30, 1-26, 1989

[Schneiderman 87] B. Schneiderman, *Designing the User Interface*, Addison Wesley, Menlo Park, CA, 1987.

[Smith *et al.* 85] R.G. Smith, H.A. Winston, T.M. Mitchell, and B.G. Buchanan, "Representation and Use of Explicit Justifications for Knowledge Base Refinement". In *Proceedings IJCAI-85*, Los Angeles, CA, pp. 673-680, August 1985.